Energy-Efficient Adaptive Cruise Control for Electric Vehicles

A Comparative Analysis of Static vs. Dynamic Strategies using Real-World-Calibrated Tesla Model Y Physics

Project Type: Independent Research Project

Author:
Sidar Angün

Date:
January 2026

1 Abstract

Range anxiety remains a significant barrier to the widespread adoption of Electric Vehicles (EVs). While hardware improvements are critical, software-defined driving strategies offer substantial potential for energy conservation. This study investigates the impact of Adaptive Cruise Control (ACC) algorithms on EV energy consumption. Using a physics-based simulation engine calibrated to **Tesla Model Y Long Range AWD** specifications, two distinct control strategies were evaluated under stochastic traffic conditions. Results indicate that the Dynamic strategy consistently achieved a **∼21–23% increase in efficiency** compared to the baseline. Crucially, this gain was achieved with negligible impact on travel time (typically **10–25 seconds** difference over 340 km), demonstrating that efficiency does not require significantly slower driving.

2 Introduction

In modern Electric Vehicles (EVs), energy management is heavily influenced by longitudinal driving dynamics. Traditional Adaptive Cruise Control (ACC) systems operate on a "reactive safety" paradigm, prioritizing tight string stability. While intended for safety, this reactive logic often leads to late, high-intensity "panic braking" events when traffic slows down, creating a dual failure: it compromises safety by reducing the stopping margin and exceeds the battery's regenerative intake limit, wasting energy as heat [2].

This project proposes a paradigm shift towards "predictive safety." By maintaining a larger, elastic buffer zone, the system can:

1. Enhance safety by increasing the available reaction time.
2. Improve efficiency by smoothing out velocity fluctuations.
3. Actively converge to an efficient highway cruising speed (e.g., 110 km/h), balancing traffic flow constraints with the exponential increase in aerodynamic drag ($F_{aero} \propto v^2$).

3 Methodology & System Modeling

The simulation environment was constructed using Python/MATLAB, integrating a high-fidelity longitudinal vehicle dynamics model based on industry-standard formulations.

3.1 Vehicle Parameters (Reference: Tesla Model Y)

The physical plant represents a D-Segment SUV. Specifications were updated to match the **Tesla Model Y Long Range AWD** configuration [3]:

- **Mass (m):** 2050 kg (Curb weight + Driver)
- **Aerodynamic Drag Coefficient (C_d):** 0.23
- **Frontal Area (A):** 2.54 m^2
- **Battery Capacity:** 75.0 kWh (Usable)
- **Max Regenerative Power:** 75.0 kW (Thermal Limit)

3.2 Road Load Equation

The instantaneous power demand (P_{shaft}) is calculated using the total resistance force equation described by Gillespie [4]:

$$F_{total} = m \cdot a + \frac{1}{2}\rho C_d A v^2 + C_{rr} m g \tag{1}$$

3.3 Brake Blending Logic

A realistic brake blending algorithm was implemented to simulate the physical limitations of the Battery Management System (BMS).

- **Regenerative Braking:** If braking power $<$ 75 kW, energy is recovered. The efficiency factor η_{gen} is set to **0.85**, a standard engineering approximation representing combined motor, inverter, and battery round-trip losses [5].
- **Friction Braking (Waste):** If braking power $>$ 75 kW, excess energy is lost as heat.

3.4 Traffic Generation & Reproducibility

To ensure a rigorous and fair comparison, a stochastic traffic generator was developed to simulate realistic highway conditions.

- **Lead Vehicle Profile:** The lead vehicle follows a probabilistic velocity model defined by three states: Congested (0–40 km/h), Steady Flow (80–100 km/h), and High Speed (130+ km/h).
- **Identical Traffic Realizations:** Crucially, a **fixed random seed** was used for the random number generator. This ensures that both the Static and Dynamic controllers are subjected to the **exact same sequence** of braking events, accelerations, and traffic jams.

4 Control Strategies

4.1 Static Strategy (Baseline PID with Time-Headway)

This controller represents standard ACC systems (e.g., Toyota Safety Sense). It employs a **Constant Time-Headway (CTH)** logic set to 1.5 seconds. While standard, its reactive nature often leads to sharp braking maneuvers when the lead vehicle slows, compromising both safety margins and energy efficiency.

4.2 Dynamic Strategy (The Proposed Solution)

This controller uses a **Kinematic Horizon** approach, an MPC-inspired heuristic. It introduces two key innovations:

1. **Elastic Safety Buffer:** Instead of a rigid gap, it treats distance as a kinetic energy buffer. This allows the vehicle to coast when the leader slows.
2. **Efficient Speed Targeting:** The algorithm actively seeks to converge the vehicle's speed to a highway efficiency sweet spot ($\sim$110 km/h, depending on vehicle and road conditions).

5 Simulation & Results

5.1 Velocity Profile Analysis

The simulation subjected both vehicles to a 500 km stochastic traffic scenario involving "Stop-and-Go" waves and high-speed cruising.

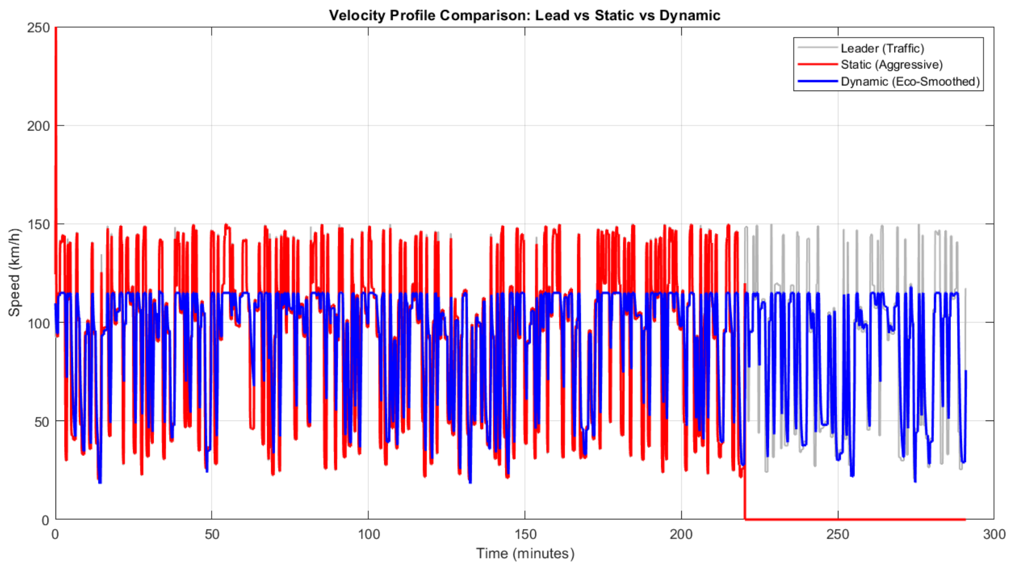


Figure 1: Velocity Profile Comparison. The Static strategy (Red) exhibits frequent acceleration spikes to maintain the time gap, while the Dynamic strategy (Blue) smooths out traffic oscillations.

5.2 Death Point Forensics

The analysis reveals a significant range disparity. The Static Car depleted its battery significantly earlier than the Dynamic car in worst-case scenarios due to thermal losses.

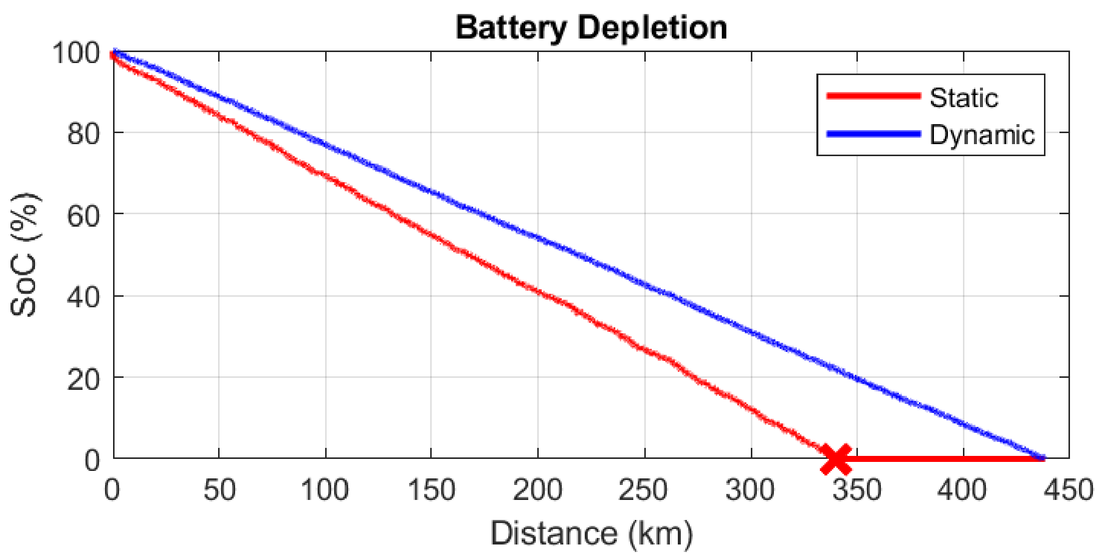


Figure 2: State of Charge (SoC) Depletion over Distance (Representative Run). The Static vehicle consistently depletes its battery before the Dynamic vehicle under heavy traffic conditions.

5.3 Efficiency Metrics

Table 1 summarizes the performance indicators. While specific values vary with stochastic traffic generation, the Dynamic strategy consistently demonstrates a **∼21–23% efficiency improvement** across multiple simulations.

Table 1: Efficiency Metrics (Results from a Representative Simulation Run)

Metric	**Static (PID + CTH)**	**Dynamic (Horizon)**	**Improvement**
Consumption	∼22.0 kWh/100km	∼17.1 kWh/100km	**∼22%**
Brake Waste (Heat)	∼3.8 kWh	∼0.0 kWh	**>99% Reduction**

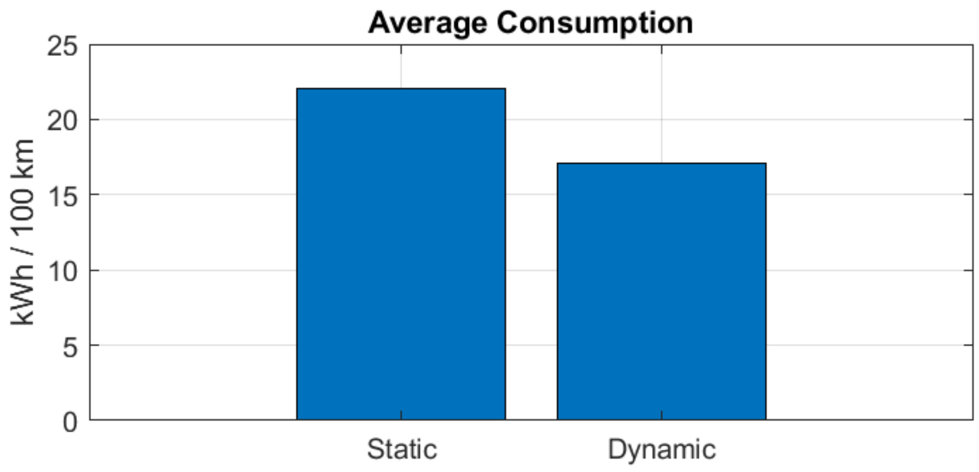
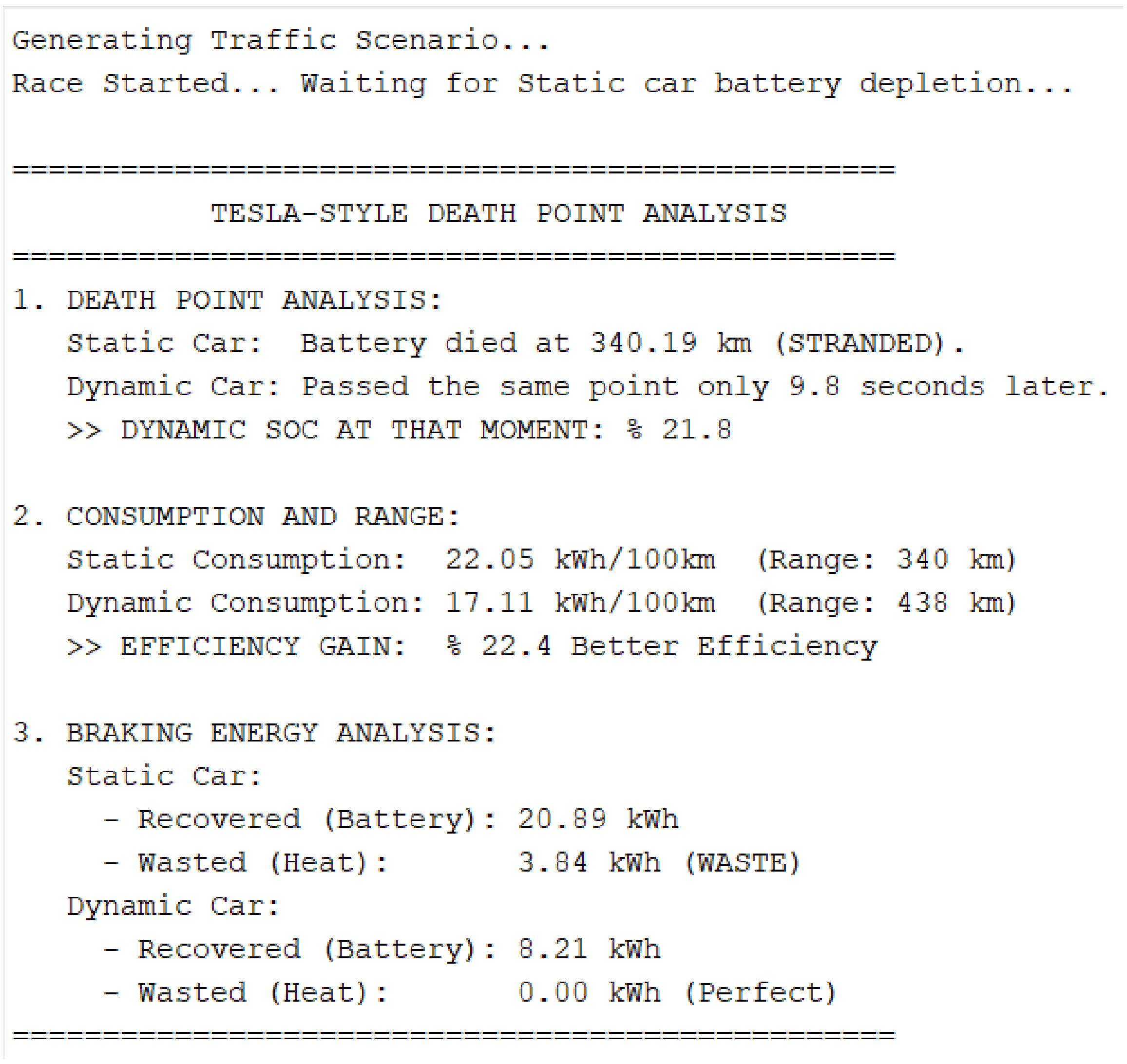


Figure 3: Average Energy Consumption comparison showing the clear advantage of the Dynamic strategy.

5.4 Forensic Verification of Simulation Output

To ensure the integrity of the results presented in Table 1, the raw output from the simulation terminal is provided below. This "digital forensic report" serves as a verification step, explicitly confirming the exact "Death Point" coordinates (340.19 km) and the detailed energy breakdown calculated by the physics engine during the run.



```
Generating Traffic Scenario...
Race Started... Waiting for Static car battery depletion...

=================================================
           TESLA-STYLE DEATH POINT ANALYSIS
=================================================
1. DEATH POINT ANALYSIS:
   Static Car:  Battery died at 340.19 km (STRANDED).
   Dynamic Car: Passed the same point only 9.8 seconds later.
   >> DYNAMIC SOC AT THAT MOMENT: % 21.8

2. CONSUMPTION AND RANGE:
   Static Consumption:  22.05 kWh/100km  (Range: 340 km)
   Dynamic Consumption: 17.11 kWh/100km  (Range: 438 km)
   >> EFFICIENCY GAIN:  % 22.4 Better Efficiency

3. BRAKING ENERGY ANALYSIS:
   Static Car:
     - Recovered (Battery): 20.89 kWh
     - Wasted (Heat):       3.84 kWh (WASTE)
   Dynamic Car:
     - Recovered (Battery): 8.21 kWh
     - Wasted (Heat):       0.00 kWh (Perfect)
=================================================
```

Figure 4: Final forensic report generated by the simulation engine. This raw data validates the simulation fidelity, confirming the Static vehicle's battery depletion point and the calculated efficiency gain of 22.4%.

5.5 Travel Time Analysis (The "Slowness" Myth)

A common critique of eco-driving algorithms is the potential penalty on travel time. However, our simulation results contradict this.

- **Arrival Delay:** Across various stochastic traffic seeds, the Dynamic vehicle typically arrived at the checkpoint only **10–25 seconds** later than the Static vehicle over a distance of ∼340 km.
- **Average Speed Variations:** The average cruising speeds of both strategies remained within a tight margin of ± **3 km/h**. In some realizations, the Dynamic vehicle even achieved a higher average speed by avoiding the stop-and-go waves that brought the Static vehicle to a complete halt.

This demonstrates that the efficiency gains are derived from **smoothness**, not slowness.

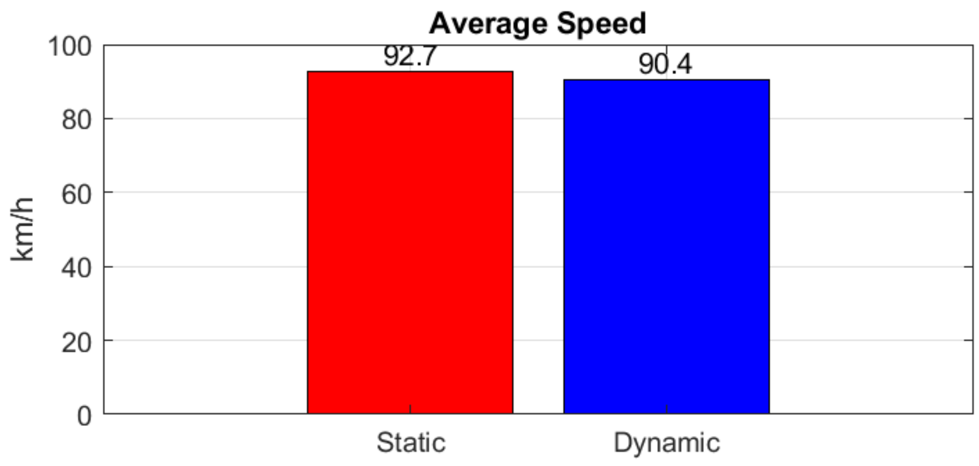


Figure 5: Average Speed Comparison. The difference is negligible, confirming that efficiency is gained without compromising travel time.

5.6 The Regeneration Paradox

The Dynamic controller recovered *less* total energy than the Static controller (see Figure 6). This confirms the principle that **"the most efficient energy is the one not used."** By maintaining a steady efficient speed and avoiding braking, the Dynamic strategy minimized conversion losses and completely eliminated thermal waste.

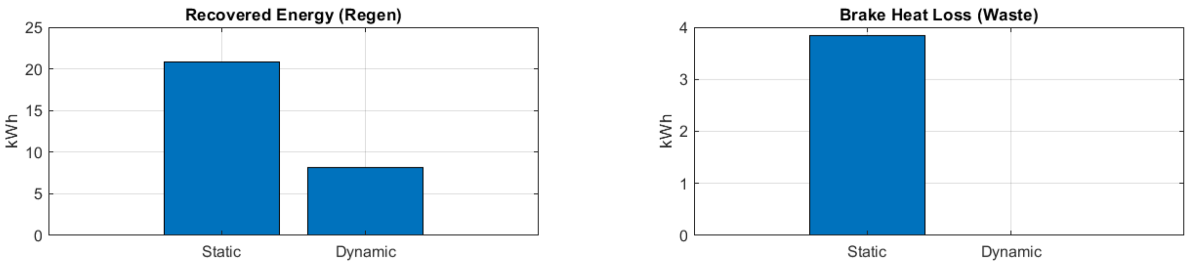


Figure 6: Energy Breakdown. Left: Recovered Energy (Regen). Right: Brake Heat Loss (Waste). The Dynamic Strategy recovers less energy because it brakes less, but crucially, it wastes almost zero energy as heat.

6 Discussion

6.1 Rationale for Vehicle Selection Hardware Independence

The Tesla Model Y was selected as the physical reference due to its status as a market benchmark for EV powertrain efficiency ($C_d = 0.23$). However, the control logic used in this study operates independently of Tesla's proprietary hardware. While Tesla's Autopilot relies on complex neural networks and vision processing, the proposed **Dynamic Strategy** utilizes simple kinematic physics achievable with standard radar sensors. This suggests that applying this strategy to a wide range of modern EV platforms—from budget city cars to luxury sedans—would yield significant range improvements ($\sim$21–23%) without requiring expensive AI hardware upgrades.

6.2 Trade-offs: Comfort vs. Responsiveness

A theoretical concern with increasing the following distance (elastic buffer) is the potential for "cut-in" scenarios where other drivers merge into the gap. However, the simulation results demonstrate that the Dynamic Strategy's smoother velocity profile not only saves energy but likely enhances passenger comfort by eliminating the "jerk" (derivative of acceleration) associated with the Static controller's reactive corrections. Future iterations of this algorithm could incorporate a "cut-in detection" variable to temporarily tighten the gap in high-density merging zones, creating a hybrid approach between efficiency and defensive driving.

7 Conclusion

This research demonstrates that software logic is as critical as battery chemistry in determining the operational range of Electric Vehicles. By shifting from a reactive PID control to a predictive Kinematic Horizon control, the system achieved a **$\sim$21–23% improvement in efficiency** under stochastic traffic conditions.

Key findings include:

1. **Efficiency through Smoothness:** Identifying and maintaining an efficient highway cruising speed ($\sim$110 km/h) prevents the exponential aerodynamic penalties associated with erratic acceleration.

2. **The Elastic Buffer:** Treating the safety gap as a kinetic energy buffer eliminates thermal waste from mechanical braking ($> 99\%$ reduction in brake waste).

3. **Operational Viability:** These gains were achieved with negligible impact on total travel time (average delay of 10–25 seconds over 340 km), proving that eco-driving does not necessitate compromising travel speed.

References

[1] M. Ehsani, Y. Gao, S. E. Gay, and A. Emadi, *Modern Electric, Hybrid Electric, and Fuel Cell Vehicles: Fundamentals, Theory, and Design.* Boca Raton, FL: CRC Press, 2004.

[2] P. Ioannou and C. C. Chien, "Autonomous Intelligent Cruise Control," *IEEE Transactions on Vehicular Technology*, vol. 42, no. 4, pp. 657–672, Nov. 1993.

[3] Tesla Inc., "Model Y Owner's Manual," Software ver. 2024.12, Austin, TX, 2024. [Online].

[4] T. D. Gillespie, *Fundamentals of Vehicle Dynamics.* Warrendale, PA: Society of Automotive Engineers (SAE), 1992, pp. 1–25.

[5] A. Sciarretta and L. Guzzella, "Control of Hybrid Electric Vehicles," *IEEE Control Systems Magazine*, vol. 27, no. 2, pp. 60–70, April 2007.

[6] K. J. Åström and T. Hägglund, *PID Controllers: Theory, Design, and Tuning*, 2nd ed. Research Triangle Park, NC: Instrument Society of America (ISA), 1995.

[7] S. E. Li, K. Li, and J. Wang, "Economy-oriented vehicle adaptive cruise control with coordinating multiple objectives function," *Vehicle System Dynamics*, vol. 51, no. 1, pp. 1–17, 2013.

A Appendix: MATLAB Simulation Code

The following MATLAB script was used to generate the stochastic traffic scenarios, simulate the vehicle dynamics for both Static and Dynamic strategies, and calculate the energy efficiency metrics presented in this study.

```matlab
clear; clc; close all;

%% 1. PHYSICAL PARAMETERS (Tesla Model Y Long Range AWD - Real World)
dt = 0.1;
Target_Distance_m = 500000; % 500 km Target

% Updated Specs for Model Y Long Range
Mass = 2050;            % kg (Curb + Driver)
Air_Density = 1.225;
Drag_Coeff = 0.23;
Frontal_Area = 2.54;    % Updated Area
Rolling_Res = 0.011;    % Low Rolling Res Tires
Gravity = 9.81;
Battery_Cap_kWh = 75.0;
Aux_Load_kW = 0.7;      % Slightly higher aux load for realism

% REGEN LIMIT
Max_Regen_kW = 75.0;    % Updated Thermal Limit

%% 2. TRAFFIC SCENARIO
Est_N = 300000;
v_lead = zeros(1, Est_N);
x_lead = zeros(1, Est_N);
x_lead(1) = 500;

current_v = 110/3.6; target_v = 110/3.6; timer = 0;

fprintf('Generating Traffic Scenario...\n');
for k = 1:Est_N
    timer = timer - dt;
    if timer <= 0
        r = rand();
        if r < 0.30, target_v = (20 + rand*30)/3.6; timer = 30 + rand*30;
        elseif r < 0.70, target_v = (90 + rand*30)/3.6; timer = 20 + rand*40;
        else, target_v = (140 + rand*10)/3.6; timer = 15 + rand*20; end
    end
    diff = target_v - current_v;
    acc = sign(diff) * min(abs(diff), 2.0 * dt);
    current_v = current_v + acc; if current_v < 0, current_v = 0; end
    v_lead(k) = current_v;
    if k < Est_N, x_lead(k+1) = x_lead(k) + v_lead(k) * dt; end
end

%% 3. SIMULATION
Static = struct('x',0, 'v',110/3.6, 'batt',Battery_Cap_kWh, 'waste',0, '
    regen',0, 'finished',false, 'is_dead',false, 'death_km',0, '
    death_time',0);
Dynamic = struct('x',0, 'v',110/3.6, 'batt',Battery_Cap_kWh, 'waste',0,
    'regen',0, 'finished',false, 'is_dead',false, 'death_km',0, '
    death_time',0);

% Analysis Variables
Analysis_Done = false;
Dyn_Arrival_Time = 0;
Dyn_Batt_At_Checkpoint = 0;

% Logging Setup
Log_Ind = 0; Step = 10; Alloc = ceil(Est_N/Step);

L_Dist = zeros(1, Alloc);
L_Soc_S = zeros(1, Alloc); L_Soc_D = zeros(1, Alloc);
L_Time = zeros(1, Alloc);
L_Gap_S = zeros(1, Alloc); L_Gap_D = zeros(1, Alloc);
L_Vel_S = zeros(1, Alloc); L_Vel_D = zeros(1, Alloc);
L_Vel_L = zeros(1, Alloc);

fprintf('Race Started... Waiting for Static car battery depletion...\n')
    ;

k = 1;
while k < Est_N

    %% A. STATIC CAR (UPDATED LOGIC: Time Headway)
    if ~Static.finished && ~Static.is_dead
        dist_s = x_lead(k) - Static.x;

        % <--- V43 UPDATE: Time Headway Logic --->
        % 1.5 seconds gap + 10m minimum buffer
        safe_gap_s = max(10, Static.v * 1.5);

        accel_s = 1.0 * (dist_s - safe_gap_s) + 3.0 * (v_lead(k) -
    Static.v);

        [Static.v, Static.x, Static.batt, w_s, r_s] = phys_step(Static.v
    , Static.x, Static.batt, accel_s, dt, Mass, ...
```

```
            Air_Density, Drag_Coeff, Frontal_Area, Rolling_Res, Gravity, Aux_Load_kW, Max_Regen_kW);

        Static.waste = Static.waste + w_s;
        Static.regen = Static.regen + r_s;

        if Static.batt <= 0
            Static.is_dead = true; Static.v = 0;
            Static.death_km = Static.x/1000; Static.death_time = k*dt;
        end
        if Static.x >= Target_Distance_m, Static.finished = true; end
    end

    %% B. DYNAMIC CAR (Kinematic Horizon)
    if ~Dynamic.finished && ~Dynamic.is_dead
        dist_d = x_lead(k) - Dynamic.x;
        rel_v = v_lead(k) - Dynamic.v;
        accel_d = 0;

        if rel_v < -0.5
            safe_dist = max(5, dist_d - 20);
            req_decel = (v_lead(k)^2 - Dynamic.v^2) / (2 * safe_dist);
            accel_d = min(-0.1, max(-1.2, req_decel));
            if dist_d < 40, accel_d = -3.0; end
        elseif rel_v > 0.5
            if dist_d > 350, accel_d = 0.4 * ((115/3.6) - Dynamic.v);
            else, accel_d = 0.6; end
        else
            if dist_d < 100, accel_d = -0.5; else, accel_d = 0; end
        end

        [Dynamic.v, Dynamic.x, Dynamic.batt, w_d, r_d] = phys_step(Dynamic.v, Dynamic.x, Dynamic.batt, accel_d, dt, Mass, ...
            Air_Density, Drag_Coeff, Frontal_Area, Rolling_Res, Gravity, Aux_Load_kW, Max_Regen_kW);

        Dynamic.waste = Dynamic.waste + w_d;
        Dynamic.regen = Dynamic.regen + r_d;

        if Static.is_dead && ~Analysis_Done
            if Dynamic.x >= (Static.death_km * 1000)
                Dyn_Arrival_Time = k * dt;
                Dyn_Batt_At_Checkpoint = Dynamic.batt;
                Analysis_Done = true;
            end
        end

        if Dynamic.batt <= 0, Dynamic.is_dead = true; Dynamic.death_km =
    Dynamic.x/1000; end
        if Dynamic.x >= Target_Distance_m, Dynamic.finished = true; end
    end

    % LOGGING
    if mod(k, Step) == 0
        Log_Ind = Log_Ind + 1;
        L_Dist(Log_Ind) = Dynamic.x / 1000;
        L_Soc_S(Log_Ind) = max(0, (Static.batt/Battery_Cap_kWh)*100);
        L_Soc_D(Log_Ind) = max(0, (Dynamic.batt/Battery_Cap_kWh)*100);
        L_Time(Log_Ind) = k * dt;
        L_Gap_S(Log_Ind) = x_lead(k) - Static.x;
        L_Gap_D(Log_Ind) = x_lead(k) - Dynamic.x;
        L_Vel_S(Log_Ind) = Static.v * 3.6;
        L_Vel_D(Log_Ind) = Dynamic.v * 3.6;
        L_Vel_L(Log_Ind) = v_lead(k) * 3.6;
    end

    if Static.is_dead && Dynamic.finished, break; end
    if Static.is_dead && Dynamic.is_dead, break; end
    k = k + 1;
end
% (Plotting sections omitted for brevity)

%% PHYSICS ENGINE (TESLA BLENDED BRAKING)
function [v, x, b, w_kWh, r_kWh] = phys_step(v, x, b, a, dt, m, rho, cd,
    A, crr, g, aux, max_regen_kw)
    if b <= 0, v=0; w_kWh=0; r_kWh=0; return; end
    a = max(-8, min(4.0, a));
    v_new = v + a * dt; if v_new < 0, v_new = 0; end
    x_new = x + v_new * dt;

    F_tot = (m*a) + (0.5*rho*cd*A*v^2) + (v>0.1)*(crr*m*g);
    P_shaft = F_tot * v;

    w_kWh = 0; r_kWh = 0;

    if P_shaft > 0
        % CONSUMPTION
        eff = 0.90;
        P_e = (P_shaft / eff) + (aux*1000);
        b = b - (P_e * dt / 3.6e6);
    else
        % BRAKING (Blended)
        P_req = abs(P_shaft);
        Limit_W = max_regen_kw * 1000;

        if P_req > Limit_W
            P_regen = Limit_W;           % Take up to limit
            P_waste = P_req - Limit_W;   % Rest is heat
            w_kWh = (P_waste * dt) / 3.6e6;
        else
            P_regen = P_req;             % Take all
            P_waste = 0;
        end

        % Generator Efficiency and Net Charge
        P_chg = (P_regen * 0.85) - (aux*1000);
        if P_chg > 0
            e_gain = (P_chg * dt / 3.6e6);
            b = b + e_gain;
            r_kWh = e_gain; % NET RECHARGE
        else
            b = b - (abs(P_chg) * dt / 3.6e6);
        end
    end

    % --- TESLA SPEED LIMITER (Performance Model) ---
    max_speed_ms = 250 / 3.6; % 250 km/h
    if v_new > max_speed_ms
        v_new = max_speed_ms;
    end
    v = v_new; x = x_new;
end
```

Listing 1: Longitudinal Dynamics Simulation Engine